

1

What's New in Windows Phone 7

WHAT'S IN THIS CHAPTER

- An architectural overview of WP7
- Comparisons with iOS and Android

Mobile application developers will find it useful to have an architectural understanding of the underlying Windows Phone 7 (WP7) software platform. In particular, developers need to understand the application framework, its capabilities and limitations, and platform extensibility. It's also important to know potential technical approaches to common development tasks. For Android or iPhone app developers, it's vital to obtain a big picture of the new WP7 operating system.



Some terms can prove confusing because different people use them in different ways. For the purposes of this book, you'll see iPhone Operating System (iOS) when the text discusses applications or code. The book uses iPhone when it asks you to perform an action with the physical device (or an emulator), such as clicking a button. In addition, we'll use WP7 and Windows Phone 7 interchangeably throughout the book.

The first chapter of this book provides an overview of the new WP7 operating system, the history of Windows Mobile, the WP7 hardware platform (also known as the chassis specification), the new Windows Compact Edition (CE) kernel, the application framework, the application store, and WP7 capabilities and limitations. Readers who aren't familiar with Windows phone technologies will see examples that use Android or iPhone technologies. It's important to understand the overall design philosophy of WP7 and its potential impact on the competition.

The chapter will outline a roadmap of Microsoft's Windows Phone offerings in the next 12-18 months. The chapter also compares the three major mobile platforms from different perspectives:

- **Underlying operating system origins:** MacOS, Linux, or Windows CE
- **Application frameworks:** Xcode on iPhone, Java on Android framework, or Silverlight and Xbox/DirectX New generation Architecture (XNA) on the WP7 app framework
- **App store process:** iPhone App Store, Android Market, or Windows Marketplace

AN OVERVIEW OF WINDOWS PHONE 7

This section will present an overview of WP7, including a brief introduction to Windows Mobile history, the design rationale of WP7 and its system architecture, as well as the new application framework and application store.

A Brief History

Microsoft's mobile operating system originated with the Pocket PC 2000 release in 2000, which was targeting Personal Digital Assistant (PDA) devices without any cellular capability. It was built on top of the Windows CE 3.0 kernel and supported multiple processor architectures, including Acorn RISC Machine (ARM), where RISC stands for Reduced Instruction Set Computer, Microprocessor without Interlocked Pipeline Stages (MIPS), and x86. The rationale was to provide scaled-down desktop experience on a mobile form factor, with a stylus mimicking the computer mouse interface, and a resistive touch screen that reacts to stylus tapping.

In 2003, Microsoft released Windows Mobile 2003 and Windows Mobile 2003 Second Edition that started to offer phone capability. This release also offered strong enterprise-oriented features such as Pocket Outlook, Virtual Private Network (VPN) support, and ActiveSync.

Then a major upgrade, Windows Mobile 5, was released in 2005. Windows Mobile 5 allowed developers to write managed applications that ran on top of the .NET Compact Framework. It also provided a Direct Push technology where Microsoft Exchange e-mails can be pushed to the Pocket Outlook client on the device as they arrive. The GUI was essentially similar to the previous releases.

Windows Mobile 6 and Windows Mobile 6.1 were released in 2007 and 2008. Both are built on top of Windows CE 5.2. The focus was still on providing a rich set of features rather than a compelling user interface (UI).

With all the Windows Mobile releases until Windows Mobile 6.1, Microsoft managed to build a strong mobile product line, targeting enterprise professionals. Its major competitor was Research in Motion (RIM). Microsoft's development efforts turned out to be quite a success from 2006 to early 2007. During this time, Windows Mobile took over 20 percent of the smartphone market and shipped 12 million devices.

The iPhone arrived in June 2007. iPhone's "Touching is believing" user experience was undoubtedly a tremendous innovation compared to any other smartphone on the market at that time. The unique multi-touch, finger-friendly user interface changed the public's opinion that smartphones were designed for professionals; as a result the smartphone market grew rapidly in the following years.

Initially Microsoft didn't realize the mobile market was undergoing a major overhaul. It failed to react quickly to accommodate the dramatic growth of the smartphone market driven by explosive adoption of the iPhone among average consumers. In 2007 and 2008, Microsoft worked on Windows Mobile 7, which for the most part resembled Windows Mobile 6 from a user interface perspective but with multi-touch support. In the interim, Microsoft released Windows Mobile 6.5, which provided a minor update with finger-friendly tiles and menus. Unsurprisingly, it failed to impress the market.

Google entered mobile space with Android in 2008, and has enjoyed rapid growth since then, partly because Microsoft has failed to release a major update for about three years (since Windows Mobile 6.1). Google has formed the Open Handset Alliance (OHA) with major handset makers, silicon vendors, and mobile operators to create the Android open platform. As Microsoft struggled to build Windows Mobile 7, handset makers turned to Google Android.

Feeling the pressure from Apple and Google, Microsoft has shuffled its mobile business division, reset the Windows Mobile 7 effort, and started WP7 from scratch. WP7 sports a new tile interface, Marketplace application store, Silverlight- and XNA-based application framework, and Xbox LIVE and Zune integration. The effort has finally paid off. WP7 was launched in Europe, Singapore, and Australia in October 2010, and in the U.S. and Canada in November 2010. Microsoft shipped 1.5 million WP7 devices in the first six weeks. It's still too early to project WP7's future in terms of market share. Nonetheless, WP7 is unique in many ways compared to iOS and Android, and thus offers another choice for smartphone users. Microsoft continues to invest in mobile technology and keeps improving Windows Phone. It'll be quite interesting to see the competition among the three major mobile operating systems for the next few years.

The Big Ideas

WP7 is the outcome of Microsoft's new mobile strategy, which is to shift from enterprise-oriented mobile product design to consumer-focused design. As Andy Lees, Microsoft's president of the mobile and embedded division, put it in an interview:

We made a very big decision to re-examine everything, because the industries surrounding mobile are at an inflection point. . . . The technological advances over the past few years enable us to do bold new things we've never done before. But the most important thing is that we are bringing it all together with an almost maniacal focus on the consumer.

www.microsoft.com/presspass/features/2010/feb10/02-15windowsphone7.mspx

The following list describes the overall goals that Microsoft tried to achieve when developing WP7:

- **Consumer Focused:** Microsoft reviewed its competitors' offerings in order to understand what the consumer wants in terms of mobile user experience. For example, consumers want to touch the screen using their fingers, rather than using a stylus. Therefore, the developer must create a graphical user interface (GUI) that's finger-friendly, with enlarged actionable components that support tapping (briefly using a finger to touch the surface), dragging

(pressing and holding an item, and moving it on the surface), flicking (briefly brushing the surface), pinching (pressing and holding, using two fingers and moving them closer), spreading (pressing and holding, using two fingers and moving them apart), and so on. In addition, WP7 applications can enable unique user experiences such as Panorama and Pivots, which are discussed in Chapter 4. Another example of consumer-focused design is the seamless integration with Microsoft's other computing assets, such as Zune media service, Xbox LIVE, Office Live, and Bing search service. This integration makes it possible for consumers to enjoy these services across different screens on different devices.

- **Life in Motion:** The rationale behind the WP7 user experience is “life in motion,” where the phone keeps pace with events happening in people's life in a well-integrated, effortless way. For example, live tiles on the Home screen show real-time updates of the user's contacts, calendars, games, messages, and phone calls. A quick glance gives the user all the needed information without the user's touching anything. And if the user touches any of those tiles, WP7 displays a hub screen where events of the selected type are aggregated into a single view from various applications, web services, and other sources.
- **Consistent experience:** The Windows Phone user experience is consistent across applications and services on assorted devices. Any third-party hardware or software innovations must be in line with the unified model to avoid fragmentation.
 - **Hardware:** Microsoft and its partners defined a set of specifications where all WP7 devices rely on a few chipsets. The reason these chipsets are so important is that Qualcomm and Microsoft have performed all the major work on the Board Support Packages (BSPs), which are driver and hardware configurations. During the Windows Mobile era, original equipment manufacturers (OEMs) had the opportunity to choose any chipset. Supporting all the chipset variants with Windows Mobile was a big challenge for Microsoft. Now, with the unique chassis specifications, a large part of the device BSPs are provided for OEMs — they only need to select some peripherals and create drivers for them. The unified hardware design is actually good for developers; there is no need to consider different CPU speeds, memory capacity, and screen sizes. They are the same on all WP7 devices.
 - **Software:** All applications are either Silverlight- or XNA-based, leveraging the same set of .NET Framework APIs. All third-party applications must pass the Marketplace certification before the user can install them on a device.

System Architecture

The WP7 OS is based on a variant of Microsoft Embedded OS, Windows CE 6 (also known as Windows Embedded CE 6), while the Windows Mobile 6.x variants are all based on Windows CE 5. Generally, Windows CE provides a 32-bit kernel that is designed for embedded devices, and a set of system services such as memory management, networking and connection management, I/O, and graphics. On the other hand, the Windows Phone OS is built on top of the CE kernel with added specific system services and an application framework for mobile phones.

The major differences between CE 6 and CE 5 are listed below:

- Process address space is increased from 32MB to 2GB. On Windows CE 5, every process can occupy 32MB of address space. Windows CE 6 increases process storage to 2GB.

- The number of processes has been increased from 32 to 32K (32,768). This is important to application developers. On Windows Mobile 6.x, only 32 processes can be active at the same time. If a user wants to launch another process, the system will reject it. WP7 eliminates the 32-process limitation.
- User mode and kernel mode device drivers are possible.
- `device.exe`, `filesystem.exe`, and `GWES.exe` have been moved to Kernel mode, to improve device performance.

Overall, the operating system in WP7 devices is more secure and stable, and offers better performance.

Figure 1-1 illustrates the WP7 OS architecture. As shown in the figure, the operating system contains three layers (from bottom to top): hardware, kernel space, and user space. All .NET Framework applications run in the user space. The OS kernel, drivers, and system services execute in kernel space. Compared to the architecture of Windows Mobile 6.5, on which you can execute both native and managed applications, WP7 OS enforces managed application development only. Furthermore, managed applications can use only the features provided by Silverlight, XNA, and Phone APIs; nothing else is accessible from within applications.

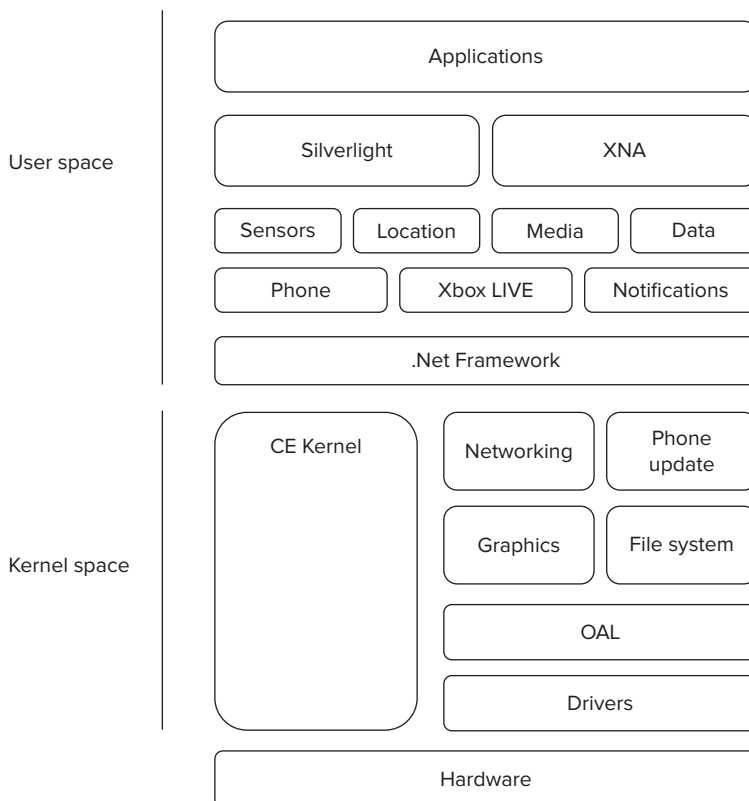


FIGURE 1-1: WP7 architecture

Application Framework

Mobile application developers are mainly concerned with changes to the application framework. When targeting Windows Mobile 6.x, developers can use either native Win32 APIs to write C/C++ code or C# and Visual Basic .NET to write managed code. The managed code runs on top of the .NET Compact Framework. On WP7, however, all applications are managed applications, and Microsoft provides two application frameworks: Silverlight and XNA, as shown in Figure 1-2 (source: [http://msdn.microsoft.com/en-us/library/ff402531\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/ff402531(v=vs.92).aspx)). Microsoft suggests using Silverlight for developing event-based applications and XNA for game development.

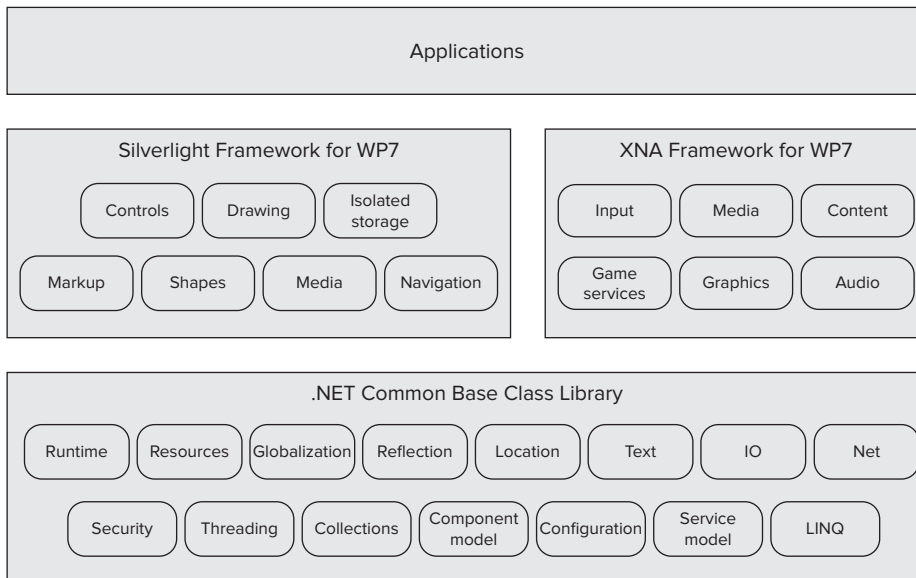


FIGURE 1-2: WP7 application framework

Some details follow on the two frameworks described briefly above (with development tools):

- **Silverlight:** People familiar with Silverlight programming techniques on the desktop will find it fairly easy to develop applications to run on WP7. Silverlight provides a .NET-based runtime environment that includes a rich user interface, multimedia, and animation. In addition, Silverlight offers web access on desktop, web server, and mobile devices. You use the Extensible Application Markup Language (XAML, pronounced “zammel”) to define the user interface, and .NET languages, such as C# and Visual Basic .NET, to implement the program logic. Silverlight on WP7 provides a subset of the .NET Framework APIs, with added phone-specific APIs. All these differences are integrated into Visual Studio. To ease the design of a rich UI, Microsoft also provides a tool called Expression Blend for UI designers.
- **XNA:** XNA is yet another .NET-based runtime environment available on Microsoft Xbox, Windows, and WP7. Microsoft optimized the XNA run time, together with extensive classes and libraries for game development. XNA provides a foundation for game developers

to create cross-platform games as long as the XNA run time runs on those platforms. You can perform XNA-based development for WP7 using C# and XNA Game Studio Express, a free tool provided by Microsoft.

- **Development Tools:** Microsoft provides Visual Studio 2010 Express for Windows Phone and Expression Blend for WP7 application development free of charge. Developers can use the built-in emulator in Visual Studio 2010 Express to debug and test an application. A Zune client is required on the desktop when you want to develop applications using a device.

Windows Phone Marketplace

Microsoft provides a Windows Phone Marketplace for obtaining WP7 applications, which is similar to the iPhone and Android offerings. Any third-party applications must pass the Marketplace certification before Microsoft will publish them in the Marketplace store. The user is able to browse store applications with the Marketplace application on the device, download free applications, or purchase paid applications. Unlike Windows Mobile 6.x, application side-loading on a retail device isn't allowed officially.

Mobile developers who want to create and publish WP7 applications and Xbox LIVE applications must register at <http://create.msdn.com/> to obtain a Marketplace developer account. The registration fee is US\$99. When a developer is ready to submit an application for certification, Microsoft validates the application's Silverlight Application Package (XAP, pronounced "zap") against published application content guidelines. If the XAP validation succeeds, Microsoft tests the application on a real device both automatically and manually to ensure it meets stability and performance policies. When all the tests are passed, the application is signed with the developer's certificate (assigned by Microsoft to the developer during Marketplace registration) and is ready for public release in the Marketplace store.

Limitations and Road Map

WP7 is a new mobile operating system that is different in many aspects from previous Microsoft Windows Mobile releases. Essentially this is a version-one product representing Microsoft's new thrust into the mobile space. Needless to say, being a version-one product implies limitations and work-in-progress to the computing world. WP7 is no exception. Some of the most frequently mentioned limitations of WP7 are described in the following:

- **Copy and paste:** This feature appears in previous Windows Mobile releases but is missing in the first release of WP7, and has been provided in a WP7 update release in March 2011. Note that iOS does not offer copy and paste until iOS 3.0. Android's full support for copy and paste starts with Android 2.3 (Gingerbread).
- **Multi-tasking for third-party applications:** While system services such as music playback and FM radio will continue to work in the background, third-party applications cannot. This is similar to the first version of iPhone. On Android, developers have the flexibility to build a service. On WP7, developers cannot create a service, and thus multi-tasking is not possible for third-party applications. It's possible that Microsoft will enable multi-tasking of a selected set of tasks, such as music playback, VoIP, and GPS, for third-party applications in a future version. However, it's unlikely that it will provide a way to write a service on WP7.

- **Native application development:** Only managed code can run on WP7. This means that third-party applications can use only those .NET classes provided by Silverlight and XNA. As a consequence, a software vendor won't be able to implement mission-critical modules in native code and access them from a managed application unless they work with a Windows Phone OEM partner directly to build the native module as an OEM extension to the operating system. (Developers can write native code modules on Android using the Native Development Kit). Microsoft is unlikely to change this design in future Windows Phone releases. The main reason for this limitation is that Microsoft wants to maintain a consistent user experience and also to make sure applications don't impact system performance and battery life.
- **HTML5 support:** The Internet Explorer web browser in WP7 does not support HTML5, whereas the browsers in Android and iOS already have solid HTML5 support because they are based on the WebKit rendering engine. Microsoft already has demonstrated Internet Explorer 9 running on WP7, which supports HTML5 in large part. It is likely that HTML5 support will be added in an update release in 2011.

Despite the limitations described in this list, there is some good news for developers: Microsoft plans to provide more frequent firmware updates to WP7 than to previous Windows Mobile releases. This means some much-needed features and flexibilities may be added to the updates.

SIDE-BY-SIDE COMPARISONS WITH ANDROID AND IPHONE

Mobile developers coming from iOS and Android will be interested in the similarities and differences between WP7, iOS, and Android. The following sections survey the three mobile operating systems from an architectural point of view.

Operating System

There are four generations of iPhones at the time of writing. Earlier iPhone models use Samsung processors, whereas iPhone 4 uses Apple's own processor, A4. All iPhones use a 3.5-inch display. The iPhone 4 has 512Mb of internal memory and either 16GB or 32GB of storage. All iPhones are equipped with GPS, Wi-Fi, and Bluetooth connectivity.

Apple derived iOS from the Mac OS X, which is Apple's desktop OS for Macintosh systems. Mac OS X is a Unix flavor combining a Mach kernel and numerous components from FreeBSD (where BSD stands for Berkeley Software Distribution) and NetBSD. iOS bears the same core OS components as Mac OS X, and Apple has optimized it for mobile devices such as iPhone, iPod Touch, and iPad with respect to performance, stability, and battery life. The iOS consists of four layers that include (from the bottom to the top): core OS layer, core service layer, media layer, and Cocoa Touch layer. iOS applications are native applications developed using the iOS SDK and Xcode development environment, and are written in Objective-C.

Android device hardware is more versatile as Google doesn't define a hardware specification for it (Microsoft does this for WP7). Generally Android devices, just like other smartphones, have cellular voice and data capability, camera, GPS, Wi-Fi, Bluetooth, and a few sensors such as an

accelerometer and a proximity sensor. Screen size varies on different devices, although 3.5-inch and 4-inch screens are becoming popular.

At the core of Android device software is the Linux kernel. The drivers and the hardware abstraction layer modules are all Linux-based. On top of the Linux kernel is the Dalvik virtual machine and its runtime environment, along with native libraries and services. Dalvik is a special Java virtual machine that's optimized for running Java applications on a mobile device. The majority of the Android operating system is the application framework, which consists of Java services, APIs, and libraries, as well as native libraries and native services. Figure 1-3 shows the details (source: <http://developer.android.com/guide/basics/what-is-android.html>).

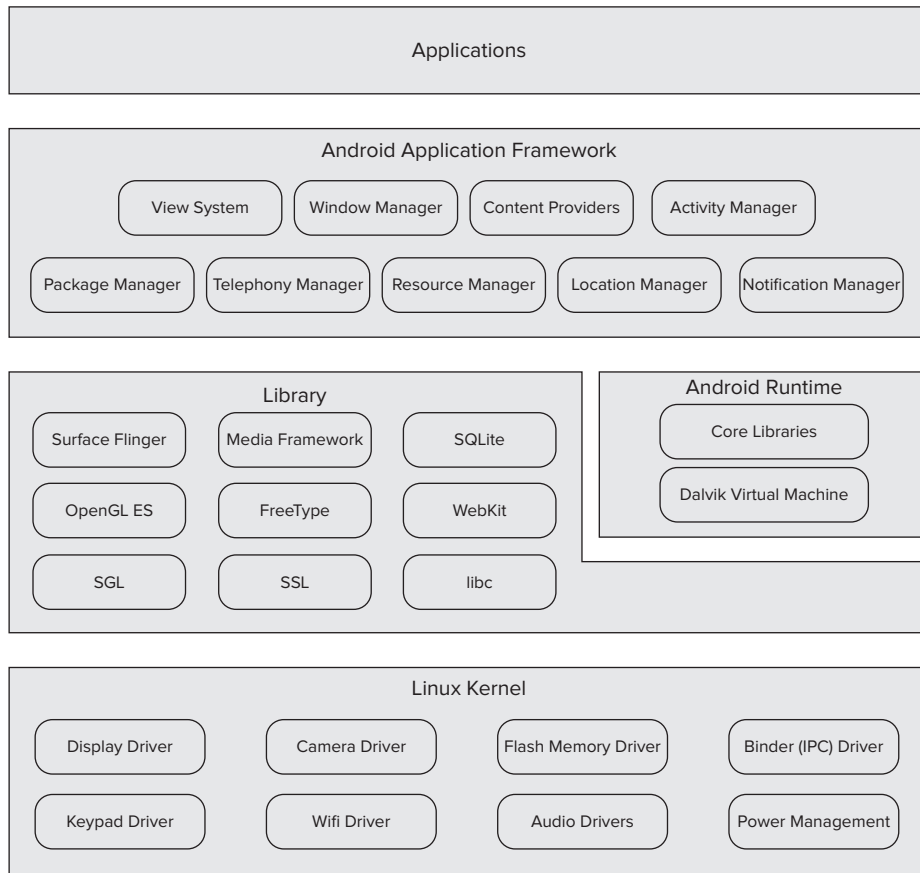


FIGURE 1-3: Android architecture

There are two application development options on the Android: Java applications that run entirely in the Dalvik virtual machine and Java applications that use JNI (Java Native Invocation) to call into native libraries on the system. You might wonder why Android developers would use a native library. Performance is the main reason, since native code generally offers better performance than Java bytecode (or dex code produced by the Dalvik virtual machine). This is particularly important

for CPU intensive code and media processing code. In addition, when someone ports C/C++ code from another platform, it's natural to wrap the existing code into native libraries instead of porting it to Java code.

On WP7, developers aren't allowed to write native code; only managed code is allowed. In addition, only subsets of the general Silverlight APIs and XNA APIs are supported on WP7. However, phone makers have the means to inject some native code into the system and use it in their applications by using COM (Component Object Model) interop. Microsoft provided a special SDK (not to be confused with the publicly available SDK for WP7 application development) for phone makers, so that they can develop native COM DLLs with a limited set of native Windows CE APIs, and use them in their applications. Those COM DLLs are treated as part of the operating system. Unfortunately, mobile developers don't have access to the special SDK.

Table 1-1 shows system level comparisons between the three operating systems.

TABLE 1-1: System Level Comparisons

ITEMS	IOS	ANDROID	WP7
Kernel	Mach kernel in Mac OS X	Linux kernel	Windows CE 6
Supported CPU architecture	ARM	ARM, X86	ARM
Memory model	Paging, no disk-backed backing store	Paging, SD-card swap space optional	Paging, no swap space
OS low-memory management	Yes	Yes	Yes
Managed run time	No	Dalvik Java run time	.NET Framework Common Language Runtime (CLR)
Application sandbox	Yes	Yes	Yes
Cross-application communication	Limited, using custom URL	Intent, Broadcast Receiver, Content Provider, and Service	Launchers and chooser

Application Framework

Understanding the differences between the application frameworks of these three major operating systems is critical to cross-platform application development.

iOS applications are built on top of the Cocoa Touch layer, which consists of Objective-C application frameworks like UIKit, Game Kit, Map Kit, and iAd. The Cocoa Touch layer provides a rich set of UI

objects and event-handling mechanisms for applications and windows, as well as core system services and media services enabled by the media layer and the core service layer. Developers will normally use the functions and methods provided by the Cocoa Touch frameworks, although it's also possible to call directly into the media and core service layers for some fine-grained control of specific services.

Similarly, Android Java applications are built on top of the Android framework that consists of an array of Java packages such as Activity Manager, View System, Windows Manager, Telephony Manager, Content Provider, Notification Manager, and so on. The UI is composed of a variety of “view” components in a combination of layouts. The framework's Java packages are the interface to the underlying system capabilities, as they rely on corresponding native libraries to perform low-level tasks. Developers can also inject native libraries into a Java application. Interaction between the Java code and the native code occurs within JNI. Android introduces some distinct programming concepts such as Activity and Intents, which makes it easier to extend system applications and services.

The WP7 application framework is .NET-based, and you can't use native code (as discussed in the “Limitations and Road Map” section of this chapter). Silverlight and XNA both provide a set of .NET Framework types that serve as the interface to the underlying operating system. In addition, applications can interact with media, local and remote data, sensors, location, and phone-specific information using the standard managed Windows Phone API. The .NET run time and libraries rely on native system services and libraries. Each application is isolated within a .NET runtime sandbox. The use of a sandbox means that cross-application communication isn't allowed on WP7, which means developers have to resort to web services as a bridge for cross-app communication.

Table 1-2 outlines application frameworks on the three platforms.

TABLE 1-2: Application Framework Comparisons

ITEM	IOS	ANDROID	WP7
Available application frameworks	Cocoa Touch	Android framework	Silverlight and XNA
Programming languages	Object-C	Java	.NET languages (C# and Visual Basic.NET)
Native API access	Yes	Yes, via JNI using NDK	No
SDK	iOS SDK on Mac OS X	Android SDK on Windows, Linux and Mac OS X	Windows Phone SDK on Windows 7
IDE	Xcode	Eclipse (with ADT plug-in)	Visual Studio 2010 or Visual Studio 2010 Express
UI definition	NIB resource file	Widgets and layouts	XAML
UI event mechanism	Delegates	Event listeners in View classes	Event handlers in CLR (Common Language Runtime)

Some developers want to be able to use C# and Visual Studio tools to develop mobile applications running on Android and iOS. This will make cross-platform design easier and enable code reuse across these platforms. Mono-Android (<http://mono-android.net/>) and Monotouch (<http://monotouch.net/>) are such tools for this purpose.

Application Store Process

The Apple App Store was the first mobile application store, and had the largest number of applications at the time of writing. Nonetheless, many mobile developers have questioned Apple's application review process because Apple has imposed so many restrictions and is known to hand out mysterious rejections. Each iOS app that a developer submits is verified against a functionality policy, content policy, and legal policy, which may take days to even a few weeks.

The Android Market grew rapidly as the number of shipped Android devices increased dramatically between 2009 and 2010. For instance, in December 2010 Google announced that it was activating 300,000 Android devices a day (<http://mashable.com/2010/12/09/android-device-stats-2010>). The Android Market enforces content policies that disallow hate speech, nudity, sexually explicit material, and copyright infringements. In contrast to the Apple App Store, the basic idea of the Android Market is openness — the Android Market doesn't perform any verification of a submitted application. Essentially, developers simply register an account, upload an application, and describe it, and the application appears in the Android Market. Google has a team that handles any reported violation against the content policy and may remove the application from the Market if necessary.



The Apple iOS content policy can be found at developer.apple.com/appstore/guidelines.html. Google Android content policy is available at www.android.com/us/developer-content-policy.html.

Microsoft started to offer a mobile application store during the Windows Mobile 6.5 years. But that effort didn't take off, mainly because the accompanying OS release failed to earn significant market share. As part of the new Windows Phone project release, Microsoft has redesigned the Windows Phone Marketplace to compare with the other two competitors. Furthermore, Microsoft seems to have learned quite a bit from the problems of the Apple App Store and the Android Market, which has resulted in a fairly comprehensive application certification process that aims at verifying a submitted application against policies and technical requirements to ensure the application is reliable, makes efficient use of resources, doesn't interfere with device functionality, and is free of malicious software. The Marketplace certification process includes both static and automatic testing of the application on API usage, stability, performance, and so on. The whole process can take as long as a few weeks.

Table 1-3 outlines the three application store processes.

TABLE 1-3: Comparison of Application Store Processes

ITEMS	APPLE APP STORE	ANDROID MARKET	WINDOWS PHONE MARKETPLACE
Registration fee	\$99 annually, \$299 annually for enterprise	One-time \$25	\$99 annually, include submissions of five free apps
App submission fee	None	None	\$19.99 for free apps; \$99 for paid
Revenue sharing (developer/company)	70/30	70/30 (Carrier takes the 30%)	70/30
Regional availability	Worldwide with country-specific stores	Free apps are accessible everywhere, but paid app access varies.	17 countries, including some EU countries, Australia, Hong Kong, India, Singapore, and U.S.
Content policy	Yes	Yes	Yes
Testing on real devices	Yes	No	Yes
Performance requirement	No	No	Yes
App return for refund	Yes	Yes, within 15 minutes after purchase	No
Try and buy	No	No	Yes

SUMMARY

After it realized that Windows Mobile wasn't the product the consumer smartphone market wanted, Microsoft decided to change its enterprise-focused strategy to a consumer-focused one. The goal of WP7 is to provide a consistent "life-in-motion" mobile experience across devices made by various handset makers.

WP7 is a completely new mobile operating system that isn't compatible with any previous Windows Mobile releases in most cases. Microsoft redesigned the GUI to provide a finger-friendly interface with multi-touch support on capacitive touch screens. The live tile Home screen is one of a kind in the industry. The integration with Xbox LIVE, Zune media service, Office Mobile, and SharePoint gives users easy access to Microsoft technologies on the go. The application framework enables developers to build Silverlight applications and XNA-based games quickly. The Windows Phone Marketplace provides an opportunity for developers to monetize their applications and games.

WP7 compares favorably with iOS and Android from various technical perspectives. The operating system kernel is Windows CE 6, which can provide 2GB of virtual address space for each process, and has significant performance and stability improvements over Windows CE 5. The application framework enables creation of a unique UI pattern and interactions with the phone functionality in a managed environment. Even if you aren't familiar with .NET technologies, you can still ramp up quickly with Windows Phone application development.

After reading this chapter, you should have the big picture of the WP7 operating system from a technical architectural perspective. In the remaining chapters, you'll learn how to leverage these innovations to build appealing applications.